The Clock-Domain-Crossing Conundrum

Jerry Cox

The Clock-Domain-Crossing conundrum is this: Where is the evidence of Clock-Domain-Crossing (CDC) failures that have occurred during a product's service life?

We know there have been products that have had CDC failures during test and been re-spun, revised or abandoned before being put into service. Theory predicts there must also be products that have had CDC failures after test. Knowledge of such failures is of utmost importance because we are becoming much more dependent on safety-critical electronics, the kind of electronics that will help you avoid colliding with that car in your blind spot, will deploy your airbag in an accident or will reduce the speed of the train on which you are riding as it negotiates a curve.

Number of CDCs per chip rapidly growing

Today's System-on-Chip (SoC) products contain a large number of independent clock domains. This is because IP cores may run at a variety of frequencies and, in addition, it is impossible to satisfy the traditional synchronous design requirement of almost simultaneous clock-signal delivery to everywhere on a large chip.

The transfer of information between clock domains, a CDC, is often the source of system design errors. An example of such an error is violation of the rule that a flip-flop input must be valid and stable during the crucial setup-and-hold interval. If this rule is not observed, the possibility exists that an input signal may be caught at an in-between level, neither high nor low, but *perhaps*; that is: perhaps it is one or the other level.

Avoiding CDC Problems

With the aid of experience and increasingly capable EDA tools, the designer can strive to avoid this condition. For independent clock domains, however, a perfect solution to the CDC problem is unrealizable. Instead, designers must make the probability of a *perhaps* vanishingly small and know their chances of encountering a perilous *perhaps* result. But before we seek the answer to that question, we need to understand:

- *WHERE* a *perhaps* might occur within a complex SoC,
- *WHEN* the circuit conditions are contributory to a *perhaps* and
- *WHETHER* the occurrence of a *perhaps* could cause a nasty system malfunction.

Increasingly, good EDA tools can provide answers to *where*. Accurate tools are also beginning to emerge that determine *when* the varying conditions of the semiconductor process, supply voltage and junction temperature can cause trouble. The answer to *whether* is more difficult. A *perhaps* signal can propagate through

control logic with an explosion of alternatives, many of which are benign, but some of the cases can be disastrous.

Describing failure mechanisms

It is usually best to assume the worst case, but the *whether* factor clouds the design experience obtained from existing SoCs. We know from anecdotal reports that CDC errors discovered during testing of SoCs have led to expensive recalls, re-spins and project abandonments. One report has been published that details such CDC errors, but for the most part companies are reluctant to discuss these failures publically.

Thus, we know that CDC design failures have been discovered during test, but how many bad designs were missed during test and then failed while the product was in service? Since the synchronizer failures in a CDC behave according to a Poisson process, we can make some projections. First, however, let's justify and characterize the use of the Poisson process model so that we can set up the CDC conundrum.

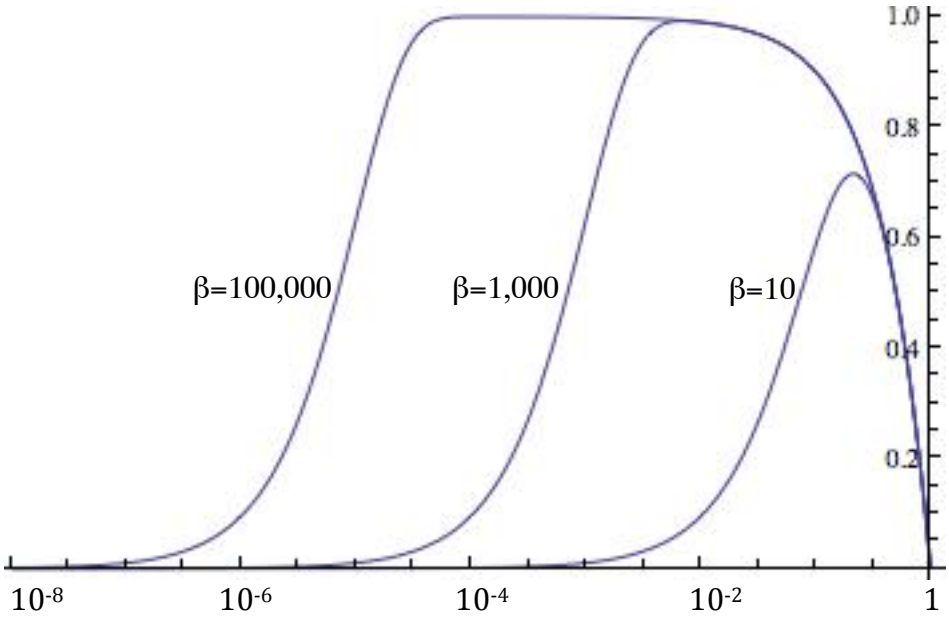
Synchronizer failures occur when the separation between clock and data signals fall in the synchronizer's window of vulnerability. Using careful design techniques, this window can be made arbitrarily small, but can never be made to vanish completely. With independent clocks at source and destination of a CDC, data arriving from the source falls uniformly throughout the destination clock period and the probability of falling in the window of vulnerability and inducing a synchronizer failure can be calculated.

Such failure events are independent of each other and the expected number of events in an interval of time depends only on the length of the interval. These conditions define a Poisson process. Note that a failure event leaves no trace behind. No lasting alteration of the synchronizer or its environment occurs as a result of a failure event. Thus, there is never any forensic evidence that points to a specific *perhaps* caused by a synchronizer failure.

Statistical Analysis

We can now describe the CDC conundrum more precisely. For a wide range of failure rates and test conditions, the Poisson process predicts the chances of failure will occur more often during service than during test. But where is the record of such failures?

The Poisson process can also predict, for a given failure rate, the chances that at least one failure will occur during the service life of all units produced, but no failure will occur during the test period. That probability for various test conditions is shown below. Even though the probability of detecting a synchronizer failure during test is small, the Poisson model predicts that, over a wide range of circumstances, an in-service failure is almost certain.



To help understand the circumstances reflected in the graph, consider an SoC that has a high probability of failure during test. The closer the probability of a test failure is to unity, the less likely there will be an in-service failure because the part will be re-spun or abandoned. In contrast, consider an SoC that has an extremely low probability of failure during test. Even so, the Poisson-process model predicts that, depending on the number of units produced and the service life; a unit can often be expected to fail in service. The parameter β shown in the graph is the ratio of the product-months in service to the product-months in test. For example, consider the case of 10,000 products in service for 100 months and 10 products in test for a month. Then the factor β = 10,000*100/(10* 1) = 100,000. Should we be worried that in-service failures are likely for test-failure probabilities that cover four orders of magnitude from 10^{-1} to 10^{-5}?

There are several possibilities that must be considered:

- The Poisson model of metastability does not apply, even though tests in silicon indicate that it does.
- A *perhaps* signal rarely, if ever, produces a nasty system malfunction.

- Serious malfunctions do occur while SoCs are in service, but are either undiagnosed or misdiagnosed because the synchronizer that produced the *perhaps* signal left no forensic evidence behind.

The third possibility seems most likely as an explanation of the CDC conundrum. Failures, when they do occur, are considered baffling or incorrectly blamed on something else. With many safety-critical SoC products currently in service or in design, it seems reasonable that steps should be taken to either verify or deny this third CDC possibility. For example, should an independent body, with access to a product's complete EDA tool flow, investigate fatal accidents involving a complex SoC? This could solve the CDC conundrum by determining which of the three above possibilities is true. Furthermore, such a step might even save lives and limit the steadily increasing liability faced by some manufacturers.

Solving the Conundrum

The take-away message is this: It is not easy to find the cause of a serious malfunction in a complex SoC. Investigators need to utilize a wide range of approaches. CDC failures are particularly difficult and the solution lies in careful analysis of an SoC design, an analysis that, because of the absence of forensic evidence, is in addition to an investigation of the individual part that failed. Thus, designers need to supply investigators with the tools and netlists that were used during the design and avoid the cloak of company secrecy that often surrounds a troubled design. Only then can the CDC conundrum be resolved and high-reliability SoCs be certified for safety-critical products.